CS534 - Machine Learning

Implementation Assignment 1

Group 50

Sebastian Mueller - 933962290

Derek Helms - 934451909

Vy Bui - 934370552

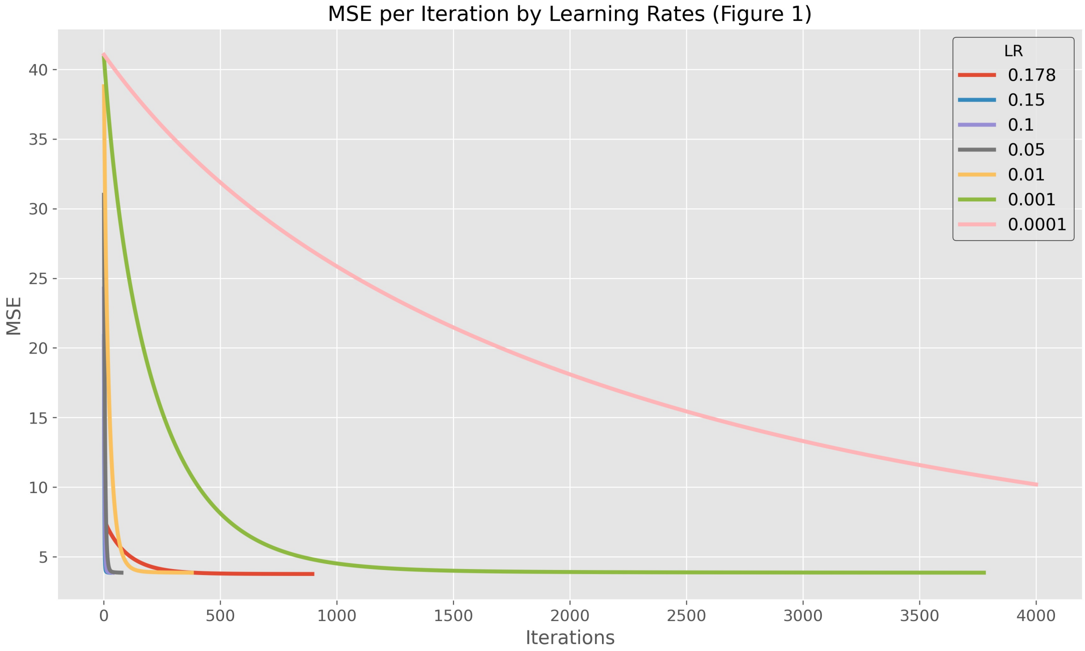
Instructor: Dr. Xiaoli Fern

The School of Electrical Engineering and Computer Science
Oregon State University

Part I

Question a: Which learning rate or learning rates did you observe to be good for this particular data set? What learning rates (if any) make gradient descent diverge?

Response:

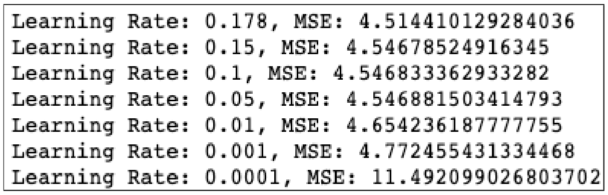


We found that using learning rates between 0.178 and 0.0001 resulted in gradient descent convergence for this data set. Anything above 0.178 was too large of a learning rate and resulted in gradient descent divergence. The idea of what actually makes a learning rate good is further expanded upon in question b.

Question b: Which learning rate leads to the best validation MSE? Between different convergent learning rates, how should we choose one if the validation MSE is nearly identical?

Response:
Using a learning rate of 0.178 resulted in the lowest MSE (a value of approximately 4.5144). If two different convergent learning rates have nearly the same MSE for the validation data, we should choose the larger learning step. This is because the two learning rates will give the model similar performance, but the larger learning rate will result in a faster training time. For our model, learning rates 0.15 and 0.05 had a difference of .0001 in their MSE. However, the model using $\eta = 0.15$ converged early in 882 iterations, while the model utilizing $\eta = 0.05$ converged early in 2646 iterations. The extra iterations for the second model ($\eta = 0.05$) would be unnecessary since

the MSE difference between them is minimal; it is not worth the time and cost off of unnecessary training.

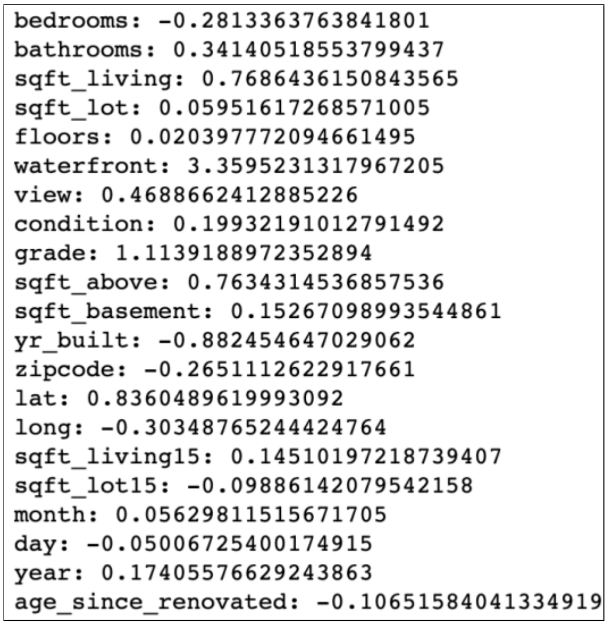


```
Learning Rate: 0.178, MSE: 4.514410129284036
Learning Rate: 0.15, MSE: 4.54678524916345
Learning Rate: 0.1, MSE: 4.546833362933282
Learning Rate: 0.05, MSE: 4.546881503414793
Learning Rate: 0.01, MSE: 4.654236187777755
Learning Rate: 0.001, MSE: 4.772455431334468
Learning Rate: 0.0001, MSE: 11.492099026803702
```

Figure 2: Different learning rates and corresponding MSE.

Question c:
learned feature weights are often used to understand the importance of the features. What features are the most important in deciding the house prices according to the learned weights?

Response:
We can use our learned feature weights as evidence for inference about features and their importance to house prices. Using our best fit model ($\eta = 0.178$) we found that waterfront, grade and yr_built had the three heaviest weights respectively (Figure 3). Waterfront and grade both increasing value while yr_built deceases value. Additionally, it is important to note that the waterfront feature proportionally had a very large weight; over 100% larger than the next largest feature (grade). This showcases how waterfront may hold a very large influence on determining house prices compared to any other features.



```
bedrooms: -0.2813363763841801
bathrooms: 0.34140518553799437
sqft_living: 0.7686436150843565
sqft_lot: 0.05951617268571005
floors: 0.020397772094661495
waterfront: 3.3595231317967205
view: 0.4688662412885226
condition: 0.19932191012791492
grade: 1.1139188972352894
sqft_above: 0.7634314536857536
sqft_basement: 0.15267098993544861
yr_built: -0.882454647029062
zipcode: -0.2651112622917661
lat: 0.8360489619993092
long: -0.30348765244424764
sqft_living15: 0.14510197218739407
sqft_lot15: -0.09886142079542158
month: 0.05629811515671705
day: -0.05006725400174915
year: 0.17405576629243863
age_since_renovated: -0.10651584041334919
```

Figure 3: Different learned feature weights for ($\eta = 0.178$) model.

Part II

Question a:
list all the learning rates that you experimented with for this part and converging or diverging status, the best learning rate and MSE of training and validation sets, and learned feature rates using best feature weight.

Response:
The learning rates we chose to experimented with were and the convergent/divergent status is shown in Figure 4. We tried learning rates up to 1e-10 and found they all diverged. The optimal range in which learning rates began to converge was between 1e-10 and 1e-11. The MSE seemed to hit an uper bound at 1e-11, beyond this threshold, MSE began to grow. The best learning rate we found was 5.5e-11 with MSE for training and validation sets of 7.714268 and 9.429637 brespectively. The learned feature weights for this learning rate are reported in Figure 5.

η	.1	.01	.001	[.0001 ... 1e-10)	(1e-10 ... 1e-11]	1e-11 and above
conv or div	div	div	div	div... div	conv ... conv	conv, but with higher MSE

Figure 4: experimented learning rates and whether they converged (conv) or diverged (div)

Question a1:
Why do you think the learning rate needs to be much smaller for the non-normalized data?

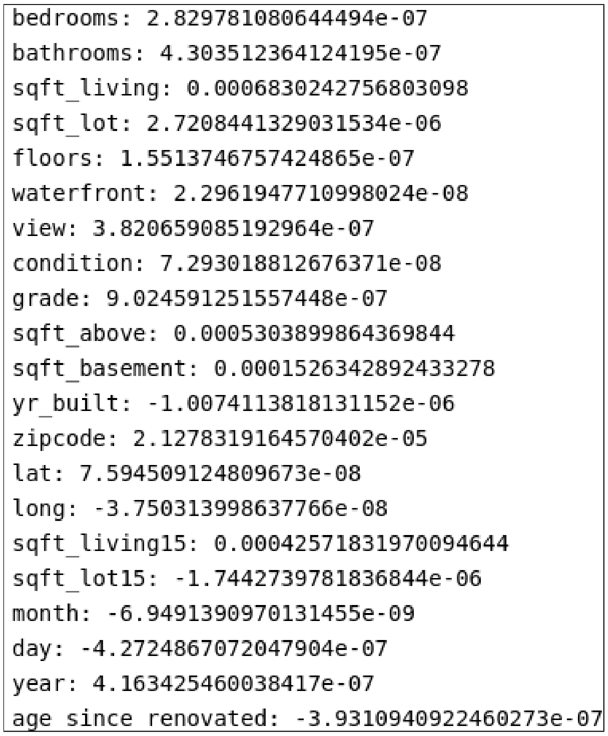
Response:
Using non-normalized data will usually result in training with feature inputs of different scales. This in turn affects our loss surface, making the contour steeper. With a steeper contour, gradient decent is far more sensitive to over-fitting; steepness of our loss curve bounds η. Normalizing data will put all features in the same scale, reducing the steepness of our loss surface/curve and result in the ability to use larger η without risk of over fitting.

Question a2:
Compare between using the normalized and the non-normalized versions of the data, which one is easier to train and why?

Response:
When comparing normalized vs non-normalized data, normalized data was easier to train. With normalized data it was easier to find an optimal η and training generally took less time; as explained above, normalization makes the shape of the loss function ”easier” and more efficient to preform gradient decent upon.

Question a3:
Compare the learned weights to those of 1(c), what key difference do you observe? How do you think this impacts the validity of using weights as the measure of feature importance?

Response:
One key difference is the spread. Our weights of normalized data features span a smaller range of lower values compared to our weights of non-normalized data features. This difference in spread highlights how feature weight magnitude observed on its own is not particularly indicative of feature importance. When we normalize the data, we are then viewing it all on the same scale and can make better assumptions about feature weight and its importance in the model. For example, waterfront no longer has the highest value with non normalized data, assuming that lat holds the most model influence would be incorrect due to these differing scales



```
bedrooms: 2.829781080644494e-07
bathrooms: 4.303512364124195e-07
sqft_living: 0.0006830242756803098
sqft_lot: 2.7208441329031534e-06
floors: 1.5513746757424865e-07
waterfront: 2.2961947710998024e-08
view: 3.820659085192964e-07
condition: 7.293018812676371e-08
grade: 9.024591251557448e-07
sqft_above: 0.0005303899864369844
sqft_basement: 0.0001526342892433278
yr_built: -1.0074113818131152e-06
zipcode: 2.1278319164570402e-05
lat: 7.594509124809673e-08
long: -3.750313998637766e-08
sqft_living15: 0.0004257183197009464
sqft_lot15: -1.7442739781836844e-06
month: -6.9491390970131455e-09
day: -4.2724867072047904e-07
year: 4.163425460038417e-07
age since renovated: -3.9310940922460273e-07
```

Figure 5: Learned feature weights for ($\eta = 5.5e - 11$) model.

Question b:
how does this new model compare to the one in part 1 (c)? What do you observe when comparing the weight for sqrt living in both versions? Consider the situation in general when two features x1 and x2 are redundant, what do you expect to happen to the weights (w1 and w2) when learning with both features, in comparison with w1 which is learned with just x1? How does this phenomenon influence the interpretation of feature importance?

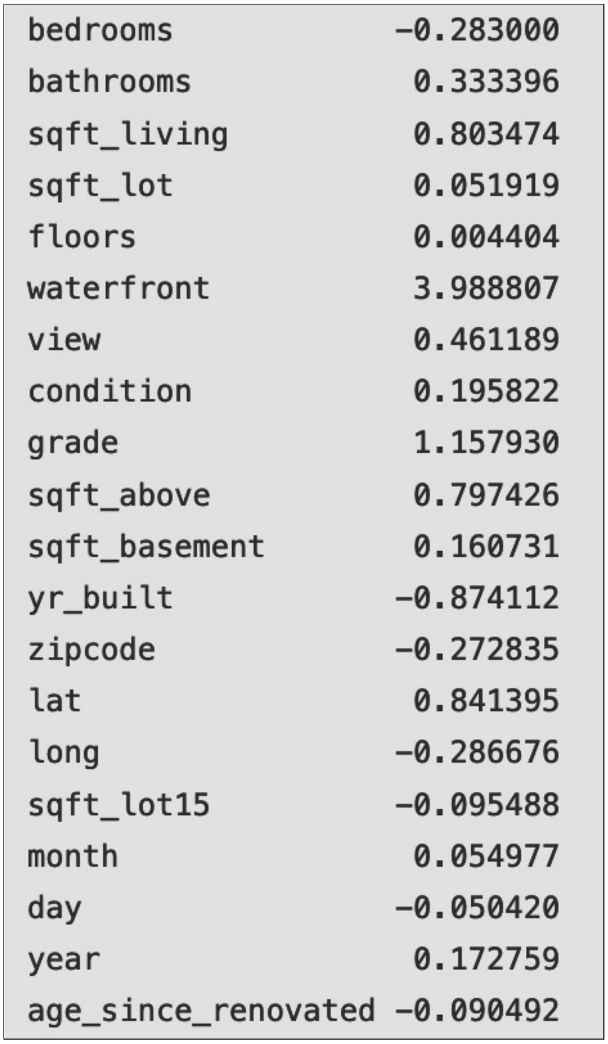
Response:
Learned feature weights for our retrained data without sqft_living 15 are reported in Figure 6. The validation MSE was 4.51934. Additionally, we see increases in magnitude for other variables when removing sqft_living 15, like waterfront.

The weight of sqft_living will be higher without including sqft_living15 in the model. During our learning step, think of the gradient with respect to sqft_living15 as a vector whose projection on the gradient with respect to sqft_living is non-zero (because they are correlated). We can decompose it into the projection plus another vector. When we use both features, the importance of the "underlying feature" (the correlated part of them) is split and carried by two feature weights. Therefore, when we only used one feature, its weight will carry the entire importance of the "underlying feature".

This example highlights how ”blind” interpretation of feature importance within a model may be misguided. It is pertinent to isolate any inter-feature dependencies and change your model accordingly before making assumptions about feature importance.

Discussion

The following is a brief discussion of our results of part I and II. We were surprised by a few things through out our implementation; however, they were either cleared up when thinking about the report questions, or through group discussions. For example, we were surprised at first that we needed to use such small learning rates with non normalized data, but then understood its relation to gradient decent. We were also surprised about how much larger the MSE was for non-normalized data, but then realized scaling affects these values and that MSE is squared, so values may be higher than expected. Lastly, there was one instance were we received a significantly worse MSE for the validation set after optimizing η, we then talked about why this did not make sense, to then discover we had a bug in our code! Other than those small bumps, our results seemed reasonably close to what we were expecting.



```
bedrooms            -0.283000
bathrooms            0.333396
sqft_living          0.803474
sqft_lot             0.051919
floors               0.004404
waterfront           3.988807
view                 0.461189
condition            0.195822
grade                1.157930
sqft_above           0.797426
sqft_basement        0.160731
yr_built            -0.874112
zipcode             -0.272835
lat                  0.841395
long                -0.286676
sqft_lot15          -0.095488
month                0.054977
day                 -0.050420
year                 0.172759
age_since_renovated -0.090492
```

Figure 6: Learned feature weights for model without sqft_living 15 feature.